IN THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A computer executed ~~implemented~~ process using memory and at least one processor for making a bytecode of a method, the bytecode of the method being stored on said memory, and said bytecode of the method made sharable by a first class loader and a second class loader, the first class loader and the second class loader being capable of dynamically loading a class having a class file, the first class loader being capable of translating the class file into a first class type and the second class loader being capable of translating the class file into a second class type, the process comprising:

dividing a runtime representation of the first class type into a first loader independent part and a first loader dependent part,

determining whether a runtime representation of the second class type can use the first loader independent part of the runtime representation of the first class type; and

if the first loader independent part of the runtime representation of the first class type can be used by the runtime representation of the second class type,

generating a second loader dependent part of the runtime representation of the second class type using the first loader independent part of the runtime representation of the first class type; and

performing a loader re-entrant interpretation of a bytecode of the method if the method is invoked, and prefixing an implementation of the bytecode of the method with a class initialization barrier upon a first use of the class, prefixing an implementation of the bytecode of the method with a link resolution barrier upon the first use of a symbolic link, and accessing one of a first loader dependent data and a second loader dependent data of the method of one of the first loader independent runtime representation and the second loader independent runtime representation being executed.

2. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 1, wherein if the first loader independent part of the runtime representation of the first class type cannot be used by the runtime representation of the second class type, the process further comprises:

generating from the second class file a second loader dependent part of the runtime representation of the second class type and a second loader independent part of the runtime

representation of the second class type.

3. (Cancel)

4. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 1 [[3]], wherein if the implementation of the bytecode requires the class initialization barrier, a need does not exist to prefix the implementation of the bytecode of the method with the link resolution barrier.

5. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 1 [[3]], wherein the operation of prefixing the implementation of the bytecode of the method with the class initialization barrier comprises:

initializing each entry of a plurality of entries of the constant pool cache of a class type being used by the bytecode upon creation of the constant pool cache of the class type if the bytecode implementation requires the class initialization barrier with a distinguishable marker;

comparing a value stored at a particular entry of the plurality of entries of the constant pool cache with the distinguishable marker when the particular entry is executed by the bytecode;

calling a runtime function configured to perform an initialization of the class type; and

replacing the distinguishable marker with data required for the implementation of the bytecode requiring the link resolution barrier.

6. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 5, wherein the distinguishable marker used for the bytecode manipulating a static variable is a null pointer.

7. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 6, wherein the null pointer is replaced with an address to the static variable upon completion of the initialization of the class.

8. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 5, wherein the distinguishable marker used for the bytecode allocating a new instance of the class type is a null value (0).

9. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 8, wherein the null value is replaced with a size of the instance of the class type upon completion of the initialization of the class type.

10. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 6, wherein the distinguishable marker used for the bytecode invoking a static method is a null pointer.

11. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 10, wherein the null pointer is replaced with a reference to the loader dependent part of the runtime representation of an invoked static method upon completion of the initialization of the class.

12. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 1 [[3]], wherein the operation of prefixing the implementation of the bytecode with the link resolution barrier comprises:

initializing each entry of a plurality of entries of a constant pool cache of a class type with a distinguishable marker upon creation of the constant pool cache of the class type;

comparing a value stored at a particular entry of the constant pool cache with the distinguishable marker when the particular entry is used for interpreting bytecode;

calling a runtime function configured to perform a resolution of the symbolic links; and

replacing the distinguishable marker with data computed during link resolution of the bytecode requiring the link resolution barrier.

13. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 12, wherein the distinguishable marker used for the bytecode manipulating an instance variable is a null offset.

14. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 13, wherein the null offset is replaced with an offset to the instance variable.

15. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 12, wherein the distinguishable marker used for the bytecode invoking a virtual method is a negative integer value.

16. (Currently amended) A computer executed ~~implemented~~ process as recited in claim 15, wherein the negative integer value is replaced with an index to a virtual table of the class type declaring the invoked method.

17. (Currently amended) A computer program embodied on a computer readable

medium storage for making a bytecode of a method sharable by a first class loader and a second class loader, the first class loader and the second class loader being capable of dynamically loading a class having a class file, the first class loader being capable of translating the class file into a first class type and the second class loader being capable of translating the class file into a second class type, the computer program comprising:

program instructions for dividing a runtime representation of the first class type into a first loader independent part and a first loader dependent part,

program instructions for determining whether a runtime representation of the second class type can use the first loader independent part of the runtime representation of the first class type; and

if the first loader independent part of the runtime representation of the first class type can be used by the runtime representation of the second class type,

program instructions for generating a second loader dependent part of the runtime representation of the second class type using the first loader independent part of the runtime representation of the first class type; and

program instructions for performing a loader re-entrant interpretation of a bytecode of the method if the method is invoked, and prefixing an implementation of the bytecode of the method with a class initialization barrier upon a first use of the class, prefixing an implementation of the bytecode of the method with a link resolution barrier upon the first use of a symbolic link, and accessing one of a first loader dependent data and a second loader dependent data of the method of one of the first loader independent runtime representation and the second loader independent runtime representation being executed.

18. (Currently amended) A computer program embodied on a computer readable medium storage as recited in claim 17, wherein when the first loader independent part of the runtime representation of the first class type cannot be used by the runtime representation of the second class type, the computer program further comprises:

program instructions for generating a second loader dependent part of the runtime representation of the second class type and a second loader independent part of the runtime representation of the second class type.

19. (Cancel)

20. (Currently amended) A computer program embodied on a computer readable medium storage as recited in claim 17 [[19]], wherein program instructions for prefixing the implementation of the bytecode of the method with the class initialization barrier comprises:

program instructions for initializing each entry of a plurality of entries of the

constant pool cache of a class type being used by the bytecode upon creation of the constant pool cache of the class type if the bytecode implementation requires the class initialization barrier with a distinguishable marker;

program instructions for comparing a value stored at a particular entry of the plurality of entries of the constant pool cache with the distinguishable marker when the particular entry is used for interpreting the bytecode;

program instructions for calling a runtime function configured to perform an initialization of the class type; and

program instructions for replacing the distinguishable marker with data required for the implementation of the bytecode requiring the link resolution barrier.

21. (Currently amended) A computer program embodied on a computer readable medium storage as recited in claim 17 [[19]], wherein the computer instructions for prefixing the implementation of the bytecode with the link resolution barrier comprises:

computer instructions for initializing each entry of a plurality of entries of a constant pool cache of a class type with a distinguishable marker upon creation of the constant pool cache of the class type;

computer instructions for comparing a value stored at a particular entry of the constant pool cache with the distinguishable marker when the particular entry is used for interpreting the bytecode;

computer instructions for calling a runtime function configured to perform a resolution of the symbolic links; and

computer instructions for replacing the distinguishable marker with data computed during link resolution of the bytecode requiring the link resolution barrier.